

Modelling Mobile Traffic Patterns Using A Generative Adversarial Neural Networks

Bo Ma^{1,2}, Bowei Yang³, Zitian Zhang⁴, and Jie Zhang^{1,2}

¹Department of Electronic & Electrical Engineering, The University of Sheffield, Sheffield S1 4ET, UK

²Ranplan Wireless Network Design Ltd, Papworth Everard, Cambridge CB23 3UY, UK

³School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China

⁴School of Information Science and Technology, East China University of Science and Technology, Shanghai 200237, China
Emails: bma4@sheffield.ac.uk, bowei@zju.edu.cn, ztzhang@ecust.edu.cn and jie.zhang@sheffield.ac.uk

Abstract—Modelling cellular traffic pattern plays a critical role to efficiently satisfy consumers' demand, which is skewed distributed and fast-varying. Although current methods can indicate the traffic fluctuation of each cell, it is still not enough as optimisation techniques require to know the traffic distribution inside cells. In this paper, Neural Network is used to improve the resolution to intra-cell level by modelling the hotspots of geo-tagged Twitter. This paper is based on our previous work, which has already quantified the linear relationship between Tweets and mobile traffic. Here, a similarity measurement is designed to quantify how two patterns are similar to each other. We applied this measurement on the geo-tagged Tweets and found that in one of three periods (day, evening, and night), the hotspots distributions are more similar than the other periods. Then for each period, Generative Adversarial Networks (GAN) is used to train a generator for modelling the intra-cell hotspots distribution. Such a trained generator can also continuously generate convincing artificial-data to expand the data set. The similarity measurement gives high similarity (above 0.8) between generated artificial-data and the real test data.

Index Terms—GAN, Social Network, Traffic Pattern

I. INTRODUCTION

Future cellular networks are estimated to deal with a dramatic (60x [1]) and undesirable increase of operational expenditure (OPEX) [2]. To reduce the operational expenditure while maintaining consumer loyalty, one of the solutions is to adequately adjust the resource deployment according to the modelling of traffic distribution. In detail, this distribution suggests the most-potential appearance locations of hotspots where generate high cellular traffic.

There are currently two main ideas to model the high-resolution changes in traffic distribution. The first one concentrated on using a parametric specification of a probability distribution function [3], such as α -stable [4] and Zipf distribution [5]. This idea is not complicated in implementation. The data for modelling is the cellular traffic collected from base stations or the traffic in grids described by histograms. The parameters of statistic functions are tuned manually or using

maximum likelihood [4] but still challenging to approximate many intractable probabilistic-computations [6].

The other idea of modelling traffic pattern is about using machine learning to train a generative network which learns the traffic distribution with parameter tuning by gradient descent [7]. Generative Adversarial Networks (GAN) is a typical application in this area. It executes an adversarial training process, which is a mini-max game between a discriminator and a generator. This method provides not only a pattern generator but also a discriminator to give judgement if the pattern becomes different. An example of the GAN-based pattern generation is in [8]. They modelled a city-scale spatial traffic distribution, but the resolution had reached the roof (minimum grid size $235m \times 235m$).

Nevertheless, these researches are facing the following challenges:

- Low resolution: the maximum spatial resolution is at a cell-level [4]. It is caused by the resolution of cellular data which is collected from base stations (BS).
- Operator biased: Using the operator's own data can be biased to one operator. An operator-neutral data is required to reflect the traffic characteristic from all operators.
- Temporal dimension not considered: the spatial traffic changes along with time have not been taken into consideration. Accordingly, previous traffic pattern modelling methods can only be used as a reference for BS deployment but not the real-time network optimisation.

To improve in these aspects, social network data-analytics based traffic pattern modelling emerges. In our previous work [9], we found that the geo-located Tweets are correlated to cellular traffic of different cells. And such a correlation has been modelled through regression. Consequently, Twitter data can improve the resolution of the traffic pattern to higher than cell-level (to intra-cell level) based on the accurate geo-tags [9].

Our idea is to use geo-tagged Twitter data to train a reliable generator for modelling the real-world distribution and expand data-set, so the low-resolution and operator biased problems are solved by using precise geo-tags. The temporal-dimension problem is solved by the designed toolchain that provides a convincing artificial-data generator for each temporal periods based on only a limited dataset. Note that, the Twitter dataset is an example of data which contains precise geo-locations

This work was supported in part by the EC H2020 Project Data Aware Wireless Network for Internet-of Everything (778305) and the Department of Science and Technology of Guangdong Province (agreement number 2018A050501008). Thanks to Dr Marco Fiore at CNR-IEIT for his help in this paper.

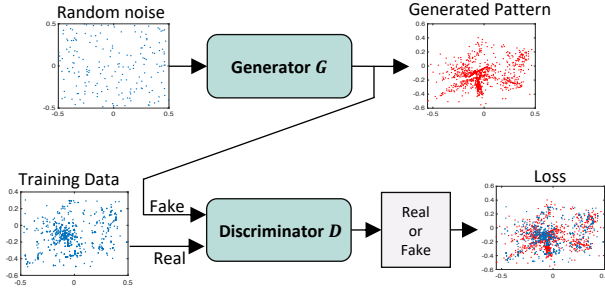


Fig. 1. The general framework of GAN [13].

and directly correlated to the network traffic. Any other data meeting these requirements can also be fed to the proposed toolchain. This design becomes useful in simulations of cellular network optimization, such as load balancing and aerial BS deployment, because the generated data is convincing, high-resolution, and temporally continuous.

In summary, this paper contributes to three aspects: The first and foremost, the temporal periods with similar distribution of coordinates can be classified into one interval. This step preprocesses the training data, so a similarity function is defined here. Then, a generator is trained by an adversarial process (GAN) for modelling the pattern of a particular interval. The trained generator continuously generates spatial-temporal hotspots with high-resolution. And the discriminator verifies if the generated data is distinct from the real data. Finally, we use the designed similarity function to quantify the reliability of the generator. The results show that generated patterns have high similarity compared to the other four distributions.

II. PRELIMINARY KNOWLEDGE

A. Generative Adversarial Networks

As stated in the introduction, the generator in GAN can be iteratively trained to model the regular hotspot distribution. This technology was first proposed by I. J. Goodfellow et al. [13] in 2014. The generator is trained to generate a fake pattern and maximise the probability to fool the discriminator with judging the fake one as a real pattern. After some iterations with convergence, the network not only models the actual pattern but also can re-generate reliable new patterns to expand the data amount. Such a traffic pattern learning is useful in network optimisation simulation to test the algorithms with reasonable and realistic patterns.

The basic framework of GAN form [13] is given in Fig. 1. The two neural networks in GAN are trained simultaneously for the generator G to learn the data distribution while the discriminator D probabilistically judges whether the generated data (from G) belongs to the true data. It is a mini-max game between two players, G and D . The training process for D is maximising the probability of classifying the generated data to fake data. In contrast, G aims at maximising the probability of D making mistakes.

B. Twitter based Mobile Traffic Demand Estimation

Our previous work [9] quantified the linear relationship between 3G traffic and number of Tweets using log-linear regression. The Twitter dataset comes from Twitter company, GNIP APIs. The London cell information comes from a UK operator, Vodafone. The regression methods optimise parameters for the fitting correlations, which can be used to formulate the model. Based on the modelled positive correlation, network traffic becomes describable using Twitter data. The estimated Down-Link (DL) traffic $\hat{r}_{DL}^{ct}$ (kbps) in cluster c and time interval t is described by corresponding number of Tweets n_{ct} (Tweets/s):

$$\hat{r}_{DL}^{ct} = 10^{b_{DL}} \left(\frac{n_{ct}}{\tau} \right)^{a_{DL}} \quad (1)$$

where $[a_{DL} = 0.88kb/\text{Tweet} \quad b_{DL} = 2.37kbps]$ and τ is ratio between time interval and second (e.g, in this work $\tau = 3600s/\text{hour}$). As a result, the real aggregate typical traffic in mobile cells can be accurately predicted for the next 2 hours using current geo-tagged Twitter activity level. The correlation is above 0.9. Besides, 71% to 79% of the variations in cellular traffic can be indicated by Twitter data. In that case, we use the geo-locations of Tweets as an indicator of mobile traffic. The spatial hotspots of Twitter are also cellular hotspots in the cells. What we need to do first is to design a similarity function and find in which period the Tweets distributions are similar.

III. SIMILARITY FUNCTION

According to [3], [7], social activities of human have periodicity, such as working, eating, or attending events. In that case, some particular places shall have users regularly arriving and leaving, such as home and metro station. The similarity means that users coordinates in different periods are close to each other. If two spatial patterns are the same, the number of coordinates should be the same, and their latitude and longitude are also the same. Only a little difference can decrease the similarity, so we choose two factors (number and change of coordinates) to quantify the similarity.

The similarity describes the tendency of two patterns nearer to each other in the temporal dimension to be more similar than patterns further apart in time [10]. There are already some applications of the similarity measurement, such as Moran's I and random sample consensus [11]. These methods have been widely applied in gesture recognition with a fixed number of points. In our view, there is still no proper method considering the similarity of the point clouds with a varying number of points. In this paper, we design a suitable similarity function to quantify the similarity between patterns in the time dimension (geo-tagged Tweets point cloud). Then, it highlights the highest-similarity period, where point cloud is defined as a set of coordinates.

We denote the two 2D point-clouds as A and B . The point coordinates in each cloud $a \in A$ and $b \in B$, and the number of coordinates are N_A and N_B . Euclidean distance matrix between a and each element in B is $||a, B||$, so the distance

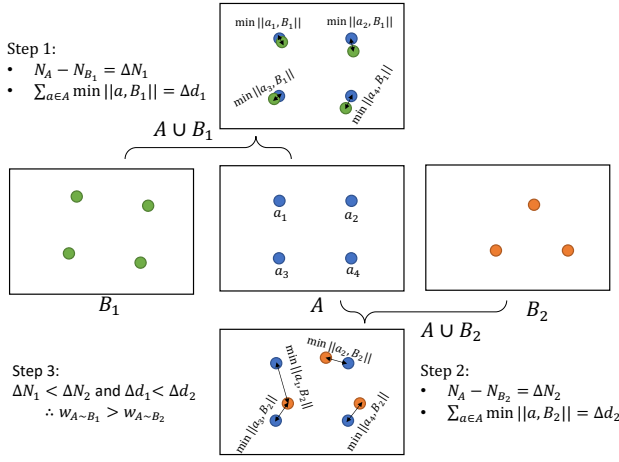


Fig. 2. An example of illustrating the process of calculating similarity.

between a and its closest b is $\min ||a, B||$. Then, the sum of minimum distance between A and B is $\sum_{a \in A} \min ||a, B||$. We present a simple example to illustrate the relationship between the similarity and the above two factors in Fig.2. It is to quantify the similarity and compare which B is more similar to A . The difference between number of points (ΔN_1 and ΔN_2) and the sum of minimum distance (Δd_1 and Δd_2) are calculated in step 1 and step 2. In the final step, it is not difficult to find that the point cloud B_1 is more similar to A than B_2 ($w_{A \sim B_1} > w_{A \sim B_2}$) because $\Delta N_1 < \Delta N_2$ and $\Delta d_1 < \Delta d_2$ (step 3).

According to the above analysis, the similarity measurement of two point-clouds A and B is defined as:

$$w = \frac{1}{1 + (k_1 |N_A - N_B| + \epsilon_1) * (k_2 \sum_{a \in A} \min ||a, B|| + \epsilon_2)} \quad (2)$$

w is the weight of similarity. The two factors ϵ and k control the significance of the number difference $|N_A - N_B|$ and minimum distance $\sum_{a \in A} \min ||a, B||$ to influence the similarity. And ϵ also helps avoid the production to be zero. In the following data analytics, a temporally ordered sequence of point clouds $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$ require the measurement of a symmetry matrix of self similarity $W = [w_{ij}]_{n \times n}$, $0 < w_{ij} \leq 1$ and $w_{ij} = w_{ji}$, diagonal elements are one ($w_{ii} = 1$). Therefore, the square symmetric similarity matrix W between pairs of pattern features is shown as follow:

$$W = \begin{bmatrix} 1 & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & 1 & w_{23} & \dots & w_{2n} \\ w_{31} & w_{32} & 1 & \dots & w_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & w_{n3} & \dots & 1 \end{bmatrix} \quad (3)$$

where n is the number of time intervals in the data set.

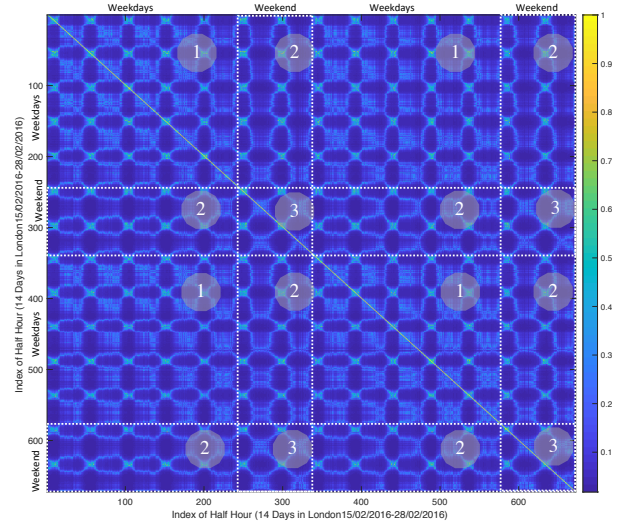


Fig. 3. The visualisation of similarity matrix of the Tweets point cloud (Tweets coordinates on map in 2016). The numbers indicate different kinds of similarities, weekdays' similarity (1), similarity between weekdays and weekends (2), weekends' similarity (3).

IV. DATA PRE-PROCESSING

A. Temporal Pre-Processing: High Similarity Period Selection

We applied the designed similarity measurement on a captured London Twitter dataset for verification. There are 0.6 million geotagged Tweets for the Greater London and surrounding suburbs area from 15/02/2016 to 28/02/2016 (total two weeks and time resolution in seconds). The similarity matrix of two weeks is visualised in Fig.3. In detail, one point cloud is the Tweets locations with coordinates distributed on the map in 0.5 hours, so there is a total of 672 frames in two weeks. The quantification of similarity is visualised as a heat level that bright pixel represents high similarity. The temporal origin is located at the top-left corner from 00:00 AM on 15/02/2016. From this result, the findings are summarised as follow (the item numbers are also marked on Fig.3):

- 1) The Fig. 3 shows alternately bright and dark rectangles which are caused by the socially in-active time of night (1:30 AM - 6:30 AM) and active time (7:00 AM - 11:00 PM).
- 2) Week-days and weekends are different. Weekends are more similar with other weekends and vice versa, which can be seen from the darker columns of weekends mapping to week-days. Therefore, it would be better to build two types of models.
- 3) The awake time of weekends is later than weekdays. For example, as shown in the similarity of weekends, the active time started at around 10:00 AM and ended at around 10:00 PM. It should be noted that this result does not mean consumers become more in-active during weekends. On the contrary, it verified that user behaviours become more random (less similarity) during weekends than weekdays, so the high-similarity period shrinks.

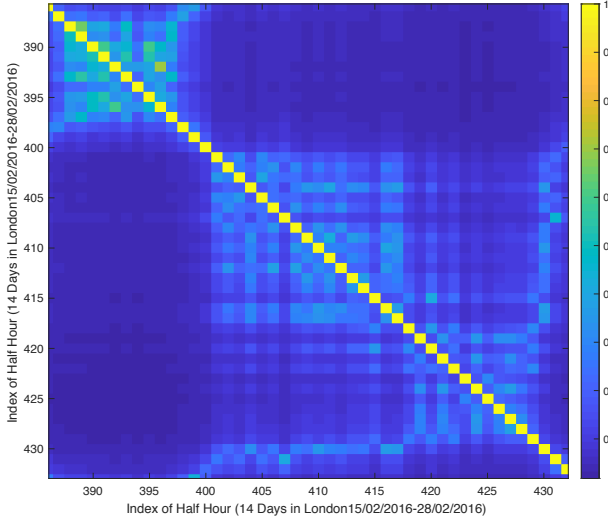


Fig. 4. One day of the similarity pattern with segmentation of night, day, and evening.

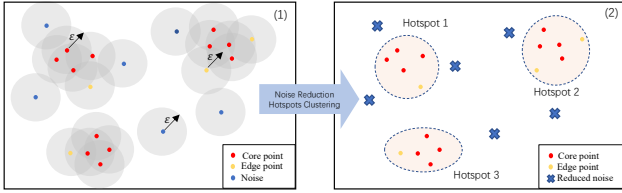


Fig. 5. The process of using DBSCAN to reduce the noise data and reserve the clusters (hotspots).

The similarity in the same day can be further divided into three blocks. For example, in Fig. 4 with zoomed-in similarity matrix, we can find three bright blocks indicating three periods in one day with high similarity, they are night (1:30 AM - 6:30 AM), day (7:00 AM - 15:30 PM), and evening (18:00 PM - 23:00 PM).

According to the above findings, we choose to use the data between 9:00 AM to 12:00 PM in the two weeks as the training data for modelling. Although the data in this period have a high similarity, noise still exists and misdirects the modelling of hotspots resulting in slow convergence. In that case, a pre-processing procedure is required to filter the noise, which will be introduced in the next part.

B. Spatial Pre-Processing: Noise Reduction by DBSCAN

To filter the noise, we choose a density-based unsupervised learning method, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) designed by M. Ester in [12]. This algorithm can automatically cluster the data in ‘hotspots’ (with many nearby neighbours) and regard the data with nearest neighbours far away as ‘noise’ so that the hotspots in the traffic distribution become the clusters generated by DBSCAN.

We briefly illustrate the procedures of DBSCAN in Fig. 5 and explain as the following steps. Firstly, as shown in the

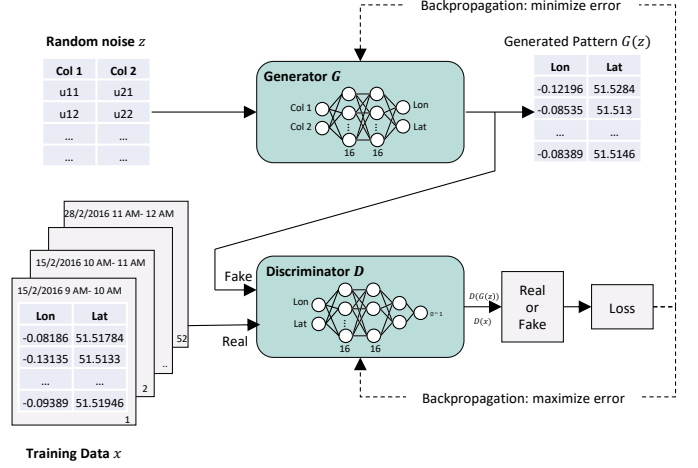


Fig. 6. The detail framework of GAN in this study.

left figure, a radius ε is visualised along with each coordinate (x, y) to provide a range (grey circles). A coordinate will be allocated to clusters (hotspots) when they have neighbouring coordinates $\geq \text{minThreshold}$ in its circle (which is 2 in this example). In contrast, the edge coordinates of clusters own neighbours but $< \text{minThreshold}$.

Moreover, as the noise coordinates hold no neighbours, they can be picked and filtered out. Then, as highlighted by dash line circles in the middle figure, the algorithm has disregarded the noise coordinates and grouped the cluster coordinates into three clusters. These clusters (hotspots) not only suit the required density but also reduce the further influence of noise to improve the performance. Finally, the cluster centroid can be produced by the mean of all the locations of points in the cluster.

This pre-processing procedure will provide the filtered data that are the cluster coordinates with a minimum density threshold. The noise data (with low density) has been ignored to provide a fast convergence of the modelling. Then, we use such a data set to train neural networks for modelling these clusters, which are the hotspots in traffic distribution.

V. GAN-BASED TRAFFIC PATTERN MODELLING

The requirements are to model the spatial distribution and generate convincing artificial data to expand the limited dataset. In machine learning, generative networks own the ability to generate artificial data. Moreover, GAN owns a discriminator to quantify how convincing the generated patterns are, so it suits our requirements.

The application framework of GAN in this paper is presented in Fig. 6. We give a case study based on the GAN designed in [13]. In detail, the training data is denoted as x . The generator G needs to model the distribution p_g over data x using input noise variables $z \sim p_z(z)$ and outputs a distribution sample $G(z)$. The discriminator D regards x and $G(z)$ as inputs and produces single scalars $D(x)$ and $D(G(z))$. The discriminator D is trained to maximise $D(x)$

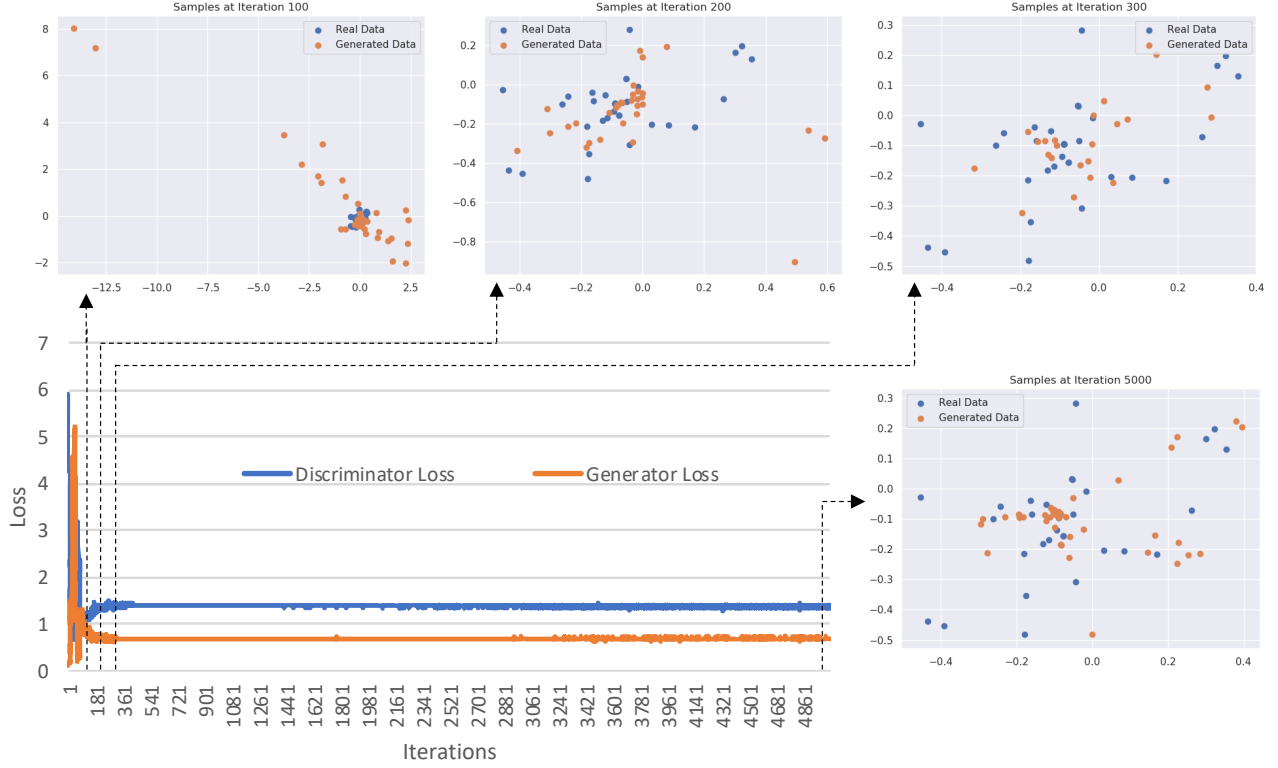


Fig. 7. Results of pattern generation.

and $\log(1 - D(G(z)))$ that higher probability comes from true data x but not the generated data $G(z)$. In this paper, G and D have two layers with 16 neurons in each layer and a Leaky ReLU activation function. Generally, D and G play the mini-max game to mutually improve their abilities of two neural networks with the following joint objective $V(D, G)$ which designed by I. J. Goodfellow et al. in [13]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_x(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]. \quad (4)$$

At the beginning of training, the noise z is fed to the generator with uniform distribution. With the comparison of real pattern x , D is trained first and can easily reject the pattern from the generator. Then, G is updated according to the loss with back-propagation. The training time in each iteration is the same for both G and D . The training process is to optimise the parameters of the two neural networks. G and D are trained iteratively (one at a time) for n_G and n_D sub-epochs until the convergence of loss functions.

The sigmoid cross-entropy in Tensorflow is used to calculate the loss. The logits (output of neural networks) is transformed by a sigmoid function to normalise the range to $[-1, 1]$. Moreover, RMSprop optimiser is chosen in the training process to find a trade-off between performance and ability of convergence. The GAN converges until D is unable to identify the difference between the real distribution and the generated distribution ($p_g = p_x$).

VI. RESULTS

In this experiment, we used MATLAB 2018b to calculate the similarity, Scikit-Learn 0.20.3 for DBSCAN, and TensorFlow 1.12.0 for training and testing GAN on Python 3.6.8. Natural logarithm is selected in the loss function (as well as objective function). As the training process converges when the discriminator can not distinguish the pattern is generated or real, the output of D is always $\frac{1}{2}$ for every input. According to Equation 4, $V(D, G)$ converges to $2\ln(\frac{1}{2}) \approx -1.39$. The line chart in Fig. 7 shows the changes of loss in each iteration, the blue line of D loss converges at around 1.39 because we dismiss the minus for convenience. Similarly, the generator loss (orange line) converges to $-\ln(\frac{1}{2}) \approx 0.69$.

A. Generated Tweets Locations

In Fig. 7, the dash-line arrows point to the patterns by comparing the generated data (orange points) with one of the real patches in the training data (blue points). Along with the convergence of the loss, the generated data is becoming acceptable after the iteration 300. The locations of real and generated coordinates are not exactly the same but indicating the common distribution to reveal the regular hotspots. Also, the patterns for training are not constant in different days, so the variance will not be averaged to each day. Currently, the generator finishes the modelling of hotspots and can be used to expand the dataset with reliable loss.

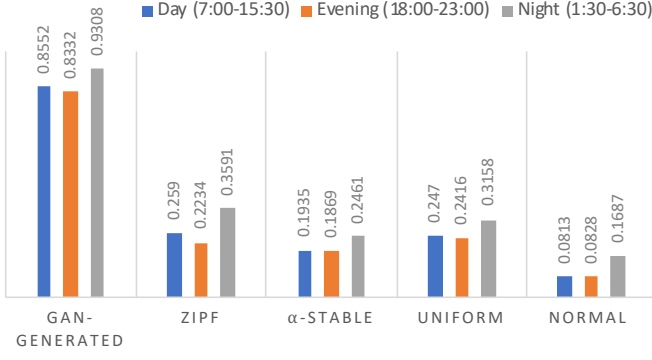


Fig. 8. Similarity comparison with the test data set.

Besides training and generating, the designed similarity measurement is also applied to the generated distribution to quantify how similar it is to the real distribution (test data). To give a reliable comparison of the performance, we also did the same procedures on the other four probabilistic distributions. The similarity comparison is shown in Fig. 8. We successively trained three generators for the three time period (day, evening, and night) during weekdays (Monday to Friday). GAN-generated patterns own the highest similarity (blue columns above 0.8) due to the similar number of coordinates and close locations. We also provide the similarity between test data and the other four probabilistic distributions. The number of points is set to be same as the testing data. In that case, the difference between distribution becomes the only factor to influence the similarity. As shown in the result, the similarity measurements (orange and grey columns) are much lower (below 0.4) because of the diverse spatial distributions. The actual Tweets distribution contains not only sparsely distributed ones but also the hot spots. Zipf and α -stable distributions gain a low performance because of lacking the ability to model sparse distribution. In contrast, uniform distribution is not good at modelling the hotspots. Note that, combinations of such probabilistic distributions will gain better performance of modelling but requires efforts for tuning the parameters. That reflects the convenience of using GAN from another aspect.

B. Mobile Traffic Pattern

The last step of this toolchain should be transforming the traffic of Tweets to the estimated traffic of cellular networks. Note that, the core function of such transformation is applying the estimation correlation. Here, Eq. 1 in preliminary knowledge is used. Thanks to the efforts of GAN, the generated locations of Tweets are coordinates so that the resolution of rasterisation is flexible. As shown in Fig. 9, the GAN-generated Tweets locations are rasterised into a spatial Tweets pattern (two-dimensional histogram). In each grid, the number of Tweets is counted and fed to the estimated correlation. Finally, the spatial traffic pattern is generated with estimated traffic in each grid. The size of the grid can be set flexibly. We provide a 5×5 example to visualise the process in a good

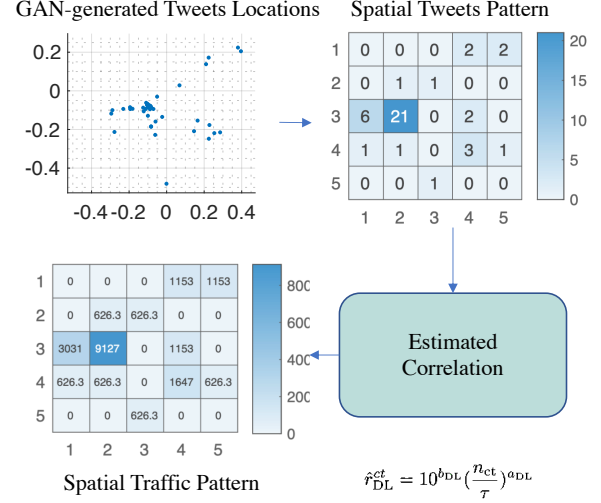


Fig. 9. Estimated cellular traffic pattern based on the generated locations of Tweets and their correlation to cellular traffic.

manner, other sizes can also be selected, such as 10×10 or 20×20 .

VII. CONCLUSION

The future cellular network optimisation requires a high-resolution (intra-cell) traffic pattern to reduce capital and operational cost while maintaining consumer loyalty. Due to this requirement, this paper leverages the positive correlation between cellular network traffic and social network to model the intra-cell Tweets traffic distribution by neural networks for further base stations deployment or network optimisation.

This work firstly designed a new similarity measurement to determine the high-similarity time period of training data. Then, the non-hotspots data are filtered using density-based unsupervised learning (DBSCAN). Finally, we trained a neural network as the generator to model the actual hotspot distribution through an adversarial process (GAN). Such a generator can be used to expand the limited dataset and continuously generate reliable traffic distribution in further network simulation.

In future, there are still some aspects requiring improvements. In the first place, the training process of GAN may meet non-convergence. The loss of discriminator will not converge, and the generator produces patterns with low similarity. This is a well-known training difficulty of GAN. It may be improved by selecting different optimiser. There is another point, the toolchain could be tested in multiple cities other than London. Even though London contains typical urban and dense urban areas, it is still not enough to represent different kinds of cities, such as in developing countries. The last but not the least, only Twitter is found with quantified correlation with cellular traffic, other possible datasets require researching efforts.

REFERENCES

- [1] A. Imran and A. Zoha, "Challenges in 5G: how to empower SON with big data for enabling 5G," IEEE Network, vol. 28, no. 6, pp. 2733, Nov 2014.

- [2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 10651082, Jun 2014.
- [3] X. Zhang, F. Zhou, J. Ning, P. Yu, and W. Li, "Hotspot localization and prediction in wireless cellular networks via spatial traffic fitting," in *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 2018, pp. 16.
- [4] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 38993912, 2017.
- [5] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, "Understanding traffic dynamics in cellular data networks," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 882890.
- [6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 26722680.
- [7] M. Xu, Q. Wang, and Q. Lin, "Hybrid holiday traffic predictions in cellular networks," in *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 2018, pp. 16.
- [8] C. Zhang, X. Ouyang, and P. Patras, "ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. ACM, 2017, pp. 363375.
- [9] B. Yang, W. Guo, B. Chen, G. Yang, and J. Zhang, "Estimating mobile traffic demand using Twitter," *IEEE Wireless Communications Letters*, vol. 5, no. 4, pp. 380383, Aug 2016.
- [10] W. R. Tobler, "A Computer Movie Simulating Urban Growth in the Detroit Region, *Econ. Geogr.*, vol. 46, p. 234, Jun. 1970.
- [11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM*, vol. 24, no. 6, pp. 381395, Jun. 1981.
- [12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, 1996.
- [13] I. J. Goodfellow et al., "Generative Adversarial Nets, *Advances in neural information processing systems*, 2014.
- [14] Mark Veillette, "STBL: Alpha stable distributions for MATLAB (<https://www.github.com/markveillette/stbl>)", GitHub. Retrieved December 23, 2019.